APPENDIX A

Sample Code for First Preferred Embodiment

```
/*=========================================================
    Method1 sample code:

    NOTE: This code is intended to demonstrate the key
    points of the implementation for method 1. It is
    intended for clarity and simplicity, so it has not
    been optimized.

    The code is written in C with two C++ extensions:
    *   C++ style comments (everything from "//" to
        the end of the line is a comment)
    *   Variables can be declared anywhere in a
        function, not just at the start of a scope.

    This code assumes:
    *   An OpenGL context to display the pixels has
        been created and is active
    *   The GL renderer supports the
        GL_TEXTURE_RECTANGLE_EXT extension
        - non power of 2 pixels. The method would work
        without the extension, but would not be as
optimal or simple.
    *   The size of the out-of-order pixel data is
        stored in sPixelDataRect.
    *   The size of each pixel is stored in
        sBytesPerPixel. This code assumes a 2 or 4
        byte pixel. The method will work with 1 byte
        pixels, but the implementation is more
        complicated.
    *   A 2D texture, the same size as the out-of-order
```

```
            pixel data has been created and is bound to the
            id stored in sTextureID.

        Written by: Mick Foley (mickf)

        Copyright:      2003 Microsoft
    ============================================================*/

    // header files that define the OpenGL data types,
    // values and functions

    #include <gl.h>      // the OpenGL header
    #include <glext.h>   // OpenGL extensions

    // local type definition

    typedef struct struct_tRect
    {
        long fTop;
        long fLeft;
        long fBottom;
        long fRight;
    } tRect;

    // static data - see the notes for more info

    static void* sPixelData_BaseAddress;
    static tRect sPixelDataRect;
    static long sBytesPerPixel;
    static int sTextureID;

    // the code
```

```
void Method1_DrawPixels( void )
{
     long stripWidth = 8 / sBytesPerPixel;
     long numRectangles =
          ( sPixelDataRect.fRight
          - sPixelDataRect.fLeft )
     / stripWidth;
     long iCounter;

     // make sure that texturing is on
     // and we have the correct texture set
     glEnable(GL_TEXTURE_RECTANGLE_EXT);
     glBindTexture(GL_TEXTURE_RECTANGLE_EXT, 1);

     // update the texture with the data
     // from the emulator VRAM
     if ( sBytesPerPixel == 2 )
     {
          glTexSubImage2D( GL_TEXTURE_RECTANGLE_EXT,
               0,
               sPixelDataRect.fLeft,
               sPixelDataRect.fTop,
               sPixelDataRect.fRight -
                    sPixelDataRect.fLeft,
               sPixelDataRect.fBottom -
                    sPixelDataRect.fTop,
               GL_RGB,
               GL_UNSIGNED_SHORT_5_6_5,
               sPixelData_BaseAddress );
     }
     else
     {
          glTexSubImage2D(GL_TEXTURE_RECTANGLE_EXT,
```

```
                0,
                sPixelDataRect.fLeft,
                sPixelDataRect.fTop,
                sPixelDataRect.fRight -
                     sPixelDataRect.fLeft,
                sPixelDataRect.fBottom -
                     sPixelDataRect.fRight,
                GL_BGRA,
                GL_UNSIGNED_INT_8_8_8_8_REV,
                sPixelData_BaseAddress);
        }

        // draw the rectangles

        // start issuing rectangle commands
        glBegin( GL_QUADS );

        // issue a rectangle for each strip
        // with the horizontal texture coords reversed

        for ( iCounter = 0; iCounter < numRectangles;
             iCounter++ )
        {
             long stripTop =     sPixelDataRect.fTop;
             long stripLeft =    sPixelDataRect.fLeft
                  + ( iCounter * stripWidth );
             long stripBottom =  sPixelDataRect.fBottom;
             long stripRight =   stripLeft + stripWidth;

             // upper left vertex -
             // upper right texture coord
             glTexCoord2i( stripRight, stripTop );
             glVertex2i( stripLeft, stripTop );
```

```
            // upper right vertex -
            // upper left texture coord
            glTexCoord2i( stripLeft, stripTop );
            glVertex2i( stripRight, stripTop );

            // lower right vertex -
            // lower left texture coord
            glTexCoord2i( stripLeft, stripBottom );
            glVertex2i( stripRight, stripBottom );

            // lower left vertex -
            // lower right texture coord
            glTexCoord2i( stripRight, stripBottom );
            glVertex2i( stripLeft, stripBottom );
        }

        // done with the rectangle draws
        glEnd();

        // finished with all the commands
        glFlush();
    }
```